Lecture 9: Cheatsheet model

Rajat Mittal

IIT Kanpur

We have discussed the functions which separate deterministic query complexity ($D(F)$) with randomized query complexity ($R(f)$) previously (iterated majority and iterated NAND). It is also known that the quantum query complexity of OR is quadratically better than the randomized query complexity of OR. In particular, we showed that $R(\text{OR}_n) = \Omega(n)$, though Grover's search [3] shows that $Q(\text{OR}_n) = O(\sqrt{n})$.

We saw in the previous lecture that $R(f) \leq D(f) = O(Q(f)^4)$. The natural question is, can quantum achieve better separation than quadratic from randomized model of computation? This was open for 30 years and was finally answered by Shalev Ben-David [2] in affirmative. Notice that there are many partial functions for which even exponential separation is known, the question is about total functions.

This function was constructed using a new technique called *cheatsheet construction*. This was used to show many other important separations too [1]. Our goal in this lecture is to understand the cheatsheet construction and show the quadratic separation.

One of the important subroutine needed for this separation is Grover search. For us, it is enough to understand the basic result of Grover search. It states that given a list of n elements (unordered), we can search for a marked element in $O(\sqrt{n})$ queries where query verifies whether the index is marked or not. Extending this, if querying each index takes m queries, then we can find the marked element with $O(\sqrt{n}m)$ queries.

Exercise 1. Convince yourself that the Grover search shows $Q(\text{OR}) = O(\sqrt{n})$.

1 Constructing cheatsheet functions

Since there are exponential separations known for partial functions, the first attempt would be to modify the partial function and hope that some sort of separation is retained. That is, the problem is still hard for randomized algorithm and still easy for the quantum algorithm. If we make the function value 0 at all undefined inputs, a quantum algorithm might find it hard to check if the input is indeed valid or not.

Another idea could be to give a witness (or a certificate) that the input is valid. Though the certificate can help the randomized algorithm too. The main idea in the cheatsheet construction is to hide the witness (using addressing function) such that randomized algorithm can't find it.

Specifically, our address bits will be replaced by the copies of the partial function, and the target bit will be replaced by the certificates for all copies of the partial function (called the cheatsheet). Now,

- a quantum algorithm will quickly find the address (it has efficient algorithm for the partial function) and then verify the certificate to make sure that each input is in the domain of the partial function;
- a randomized algorithm will not be able to find the correct cheatsheet since the number of cheatsheets is pretty large.

The strength of the method is that it does not need any information about the partial function and only needs a good separation. We (as in the original paper) will use the Forrelation function, called FR from now on. We don't need any information about it, just that for FR on n bits (FR_n), the quantum query complexity is 1 but the randomized query complexity is $\sqrt{n}$.

Before applying the cheatsheet, we would like to make sure that the certificate size is small (it needs to be verified by the quantum algorithm) for the base function. We compose FR_n with the tribes function ($\text{AND}_n \circ \text{OR}_n$). This is because tribes has small certificate and small quantum query complexity.

Let us call this the base function $\text{FAR} = \text{FR}_n \circ \text{AND}_n \circ \text{OR}_n$. Since quantum query complexity composes nicely $Q(\text{FAR}) = \Theta(n)$. You will show in the assignment that $C(\text{FAR}) = O(n^2)$.

We are planning to apply cheatsheet on FAR (a partial function), it will be useless if its randomized query complexity is small.

Randomized lower bound on the base function:

We will first show that $R(\text{FAR}) = \Omega(\sqrt{n} \cdot n^2)$. Actually, what we will show is $R(f \circ \text{OR}_n) = \Omega(R(f)n)$.

This will immediately show that $R(\text{AND}_n \circ \text{OR}_n) = \Omega(n^2)$ (also known from previous results). Since behaviour of AND and OR is similar as the inner function, we can also conclude $R(f \circ \text{AND}_n) = \Omega(R(f)n)$.

Exercise 2. Assuming $R(f \circ \text{OR}_n) = \Omega(R(f)n)$, show that $R(\text{FAR}) = \Omega(\sqrt{n} \cdot n^2)$.

We will give an outline of the proof that $R(f \circ \text{OR}_n) = \Omega(R(f)n)$, for complete details, please see [1]. The idea is to create an R-algorithm for f using the R-algorithm for $f \circ \text{OR}_n$. Notice that the algorithm for f should only use $1/n$ times the queries (on expectation) as compared to the R-algorithm for $f \circ \text{OR}_n$.

Exercise 3. Why do we only care about expected number of queries (hint: Markov inequality)?

Our target is construct an algorithm for $f \circ \text{OR}_n$ that is correct (in the randomized sense) and it takes only $\frac{R(f \circ \text{OR}_n)}{n}$ queries.

Let A be the randomized algorithm for $f \circ \text{OR}_n$. Suppose we want to compute $f(x_1, x_2, \cdots, x_m)$ (where m is the arity to f). What should we feed to A. Notice that the input to A could be visualized as a matrix with each column being the input to the respective OR. To get x_i as the output from the i-th column, we will place x_i randomly in the i-th column and keep other entries 0. Let the constructed input be z, then we output $A(z)$ as the answer to $f(x_1, x_2, \cdots, x_m)$.

You will show in the assignment that the above algorithm (say B) gives the correct answer with probability at least the success probability of A. The important part is to show that the algorithm only takes at most $\frac{R(f \circ \text{OR}_n)}{n}$ queries in expectation to the input of f.

Exercise 4. Why don't we count the number of queries to z?

We need to show that out of all queries done by B, at most a $1/n$ fraction is asking for x_i's, rest hit the spots where we have placed 0's. Let us fix a column i. Intuitively, this is true because x_i is at a random place at the column i, and if any algorithm (let alone A) is querying x_i at $\Omega(1/n)$ fraction in expectation, we can find x_i (build an algorithm for OR_n) with expected queries $o(1/n)$. This is a contradiction to the randomized query complexity of OR.

To make it more precise, say $Y_1, Y_2, \cdots, Y_m$ be the columns in z, and anytime B queries an x_i (for some i), we say that it has queried from f-input. Notice that expected number of inputs queried are $R(f \circ \text{OR}_n)$. If we pick Y uniformly at random from all Y_i's, the expected number of queries from Y should be $R(f \circ \text{OR}_n)/m$.

Given that finding f-input in a column is hard (any decision tree of depth T can only succeed with probability T/n). The probability that B finds a f-input in Y is bounded by $R(f \circ \text{OR}_n)/mn$.

On the other hand, notice that the number of f-input queries by B on the whole input is at least $R(f)$ (it is a valid randomized algorithm for f). It can be shown that the success probability of B on Y is at least $R(f)/n$. Comparing the two bounds gives us the result. Again, for details, see [1].

Exercise 5. Why did we not keep x_i at multiple places in the column of z?

We are ready to create a total function which will achieve more than quadratic speedup.

The final cheatsheet function:

The final function (Fcs) is going to be cheatsheet applied on the FAR function. Intuitively, this means that there will be around $10 \log n =: t$ copies of FAR functions to be solved to get $10 \log n$ address bits. The addresses can point to any of the n^10 cheatsheet cells, where each cell has length $tO(n^2)$ (to keep the certificates for the t copies of FAR functions).

Let $c := C(\text{FAR}) = O(n^2)$. More formally the number of input bits to the function Fcs is going to be $tn^3 + 2^t c$. The function is going to be one iff

– every input of size n^3 to the t copies of FAR function is in its domain;
– the results of these FAR functions will give a string of length t (say $F \in \{0,1\}^t$), the target specified by this address F is a cheatsheet cell, it should contain valid certificates for the t copies of the FAR function.

Exercise 6. What happens if one of the input to FAR is not in its domain?

2 Showing the separation

The idea would be that a quantum algorithm takes $tO(n)$ queries to output some value for each of the t instances of FAR. The algorithm will then verify the cheatsheet cell pointed by this address in $tO(n)$ queries (each certificate of length $O(n^2)$ will be verified in $O(n)$ queries using Grover search). Since $t = O(\log n)$, we get a quantum algorithm with $\tilde{O}(n)$ many queries.

Let us look at the details of this algorithm. I will use $\mathrm{AND}_n \circ \mathrm{OR}_n$ and Tribes interchangeably.

Quantum upper bound:

Notice that if the input is not in the domain, the quantum algorithm for FAR can answer any arbitrary value in $\{0,1\}$. Our algorithm for Fcs will obtain a string of length t after applying the quantum algorithm for FAR on the address part.

We will only answer 1 iff the target cheatsheet pointed by this string contains the correct certificate. If the input value is 1, then all t strings in the address should be in domain, and the address should point to the correct target with the correct certificate. If we answer 1 then the cheatsheet gives the correct certificate, every input is in domain and the address points to the correct cheatsheet. So it is a 1-input.

Exercise 7. Convince yourself that this algorithm works.

Since computing the t values of FAR function only takes $\tilde{O}(n)$ queries, we can verify the entire cheatsheet in similar number of queries. That means, it is enough to show that we can verify each of the certificates (in total t certificates) in $\tilde{O}(n)$ time.

We will keep each certificate in a structured manner (with $\tilde{O}(n^2)$ bits) so that the quantum algorithm can verify it in $\tilde{O}(n)$ time. The certificate will have the value of the function as the first bit, then the n bits of output from n copies of Tribes, then the $n \log n$ bit certificates for each copy of $\mathrm{AND}_n \circ \mathrm{OR}_n$. So the total length is $1 + n + nO(n \log n)$.

Exercise 8. Why is the certificate for $\mathrm{AND}_n \circ \mathrm{OR}_n$ of $n \log n$ bits?

The quantum algorithm will query the first $n+1$ bits completely and check if the input is in the domain of the FR_n function (original Forrelation function) and the value is as claimed by the address. Then we just need to check if the certificates of Tribes are correct. There are n of them, we can find if there is one which does not match by Grover search. This will take $O(\sqrt{n})$ times the queries to check one certificate of Tribes.

The 1 certificate of Tribes consists of n pointers to 1's in each of the n blocks. Again, with Grover search, we can find if any of the pointer is incorrect with $O(\sqrt{n} \log n)$ many queries. The 0 certificate can be checked with a slightly different technique.

Exercise 9. Why can't we just check if all the n pointers are 0?

We will keep the address of the block (input to OR_n) in the first $\log n$ bits. Then quantum algorithm can read all these $\log n$ bits and then check in the address part if there is a 1. Since there are n entries in a block, this can be done in $O(\sqrt{n}) + \log n$ queries.

To summarize, we can check each certificate of Tribes in $\tilde{O}(\sqrt{n})$ queries, applying Grover search recursively, we can check if they are giving the claimed input to FR_n in $\tilde{O}(n)$ queries. Whether the claimed input

is in domain of FR_n and gives the value as computed by the address can be checked explicitly with $n+1$ queries. Since this can be done for all the t copies of FAR, we get a quantum algorithm with $\tilde{O}(n)$ queries.

The nontrivial part is to put a lower bound on the performance of the randomized algorithm. Remember that $R(\mathrm{FAR}) = \Omega(n^{2.5})$. We will put the same lower bound (up to $\log n$ factor) on $R(\mathrm{Fcs})$.

2.1 Randomized lower bound on the cheatsheet function

Again, we will give an outline of the proof. Please see [1] for more details. The main intuition is: since there are a lot of cheatsheets, if the randomized algorithm can check the right cheatsheet, it must be solving at least one instance of FAR with high probability.

For this proof, we will take the lower bounding technique for $R(\mathrm{FAR})$ and convert it to a lower bound for $R(\mathrm{Fcs})$.

Exercise 10. What is the lower bounding technique for R?

Notice that Yao's min-max lemma gives probability distributions over 0 and 1 inputs (say D^0 and D^1) which are hard to separate by any deterministic (or randomized) algorithm (why?).

Let R be a randomized algorithm for Fcs. To analyze its performance, we will assume that the inputs arise from the distributions D^0 and D^1. In particular, for any $x \in \{0,1\}^t$, let $D^x = D^{x_1} \times D^{x_2} \cdots \times D^{x_t}$.

Exercise 11. For any input from D^x, what is the correct cheatsheet?

By abusing the notation, we will say that an input for Fcs comes from D^x to mean that the address bits are coming from D^x and all the cheatsheets are set to 0. For simplicity, we will assume that all 0 is not a valid cheatsheet, so that all these inputs are 0-inputs for Fcs.

To get back to intuition, R should query cheatsheet x on distribution D^x with high probability (at least $1/3$).

Exercise 12. Why? hint: if you don't query cheatsheet x, can you create a sensitive block?

On the other hand the sum over probability of querying 2^t cheatsheets on D^x is bounded by query complexity of R. That means most of the cheatsheets are not queried with high probability. In other words, all q_x's look like,

$$\begin{array}{c} q_x = (q_x)_0 \cdots (q_x)_i \cdots (q_x)_x \cdots (q_x)_1 \\ \cdots \leq \epsilon \cdots \geq 1/3 \cdots \end{array}$$

where ϵ is some small number, and 0 and 1 denote all 0 and all 1 string respectively. We would like to set it up so that there are two inputs at Hamming distance 1, $x, y \in \{0,1\}^t$, such that $(q_x)_i$ and $(q_y)_i$ are *very different*. That should allow us to distinguish between D^0 and D^1 and hence R should require a lot of queries.

Exercise 13. Intuively convince yourself about the previous statement.

Let us formalize things more. Let q_x denote the vector of length 2^t such that each of its entries $(q_x)_i$ gives the expected probability that we query cheatsheet i on inputs from D^x. Notice that i is a number less than 2^t and hence can be viewed as a binary string of length t (the address of i).

You will show in the assignment that $\sum_i (q_x)_i$ is less than the query complexity of R for all x. Since the number of cheatsheets are much bigger than this query complexity, there is at least one i such that $(q_0)_i \leq 1/6$ (fix $x = 0$). On the other hand $(q_i)_i \geq 1/3$ because i-th cheatsheet is the correct cheatsheet.

We can move from 0 to i in t steps (such that each step changes at most one bit), so there exist x and y at Hamming distance 1 such that $(q_x)_i - (q_y)_i \geq \frac{1}{6t}$. So, the randomized algorithm R can be used to distinguish D^x and D^y with probability $\frac{1}{6t}$. Just run R on D^x or D^y and check if we query the cheatsheet i.

x and y are at Hamming distance 1, so D^0 and D^1 can be distinguished with probability $\frac{1}{6t}$ using R. Remember that D^0 and D^1 are hard distributions for $R(\mathrm{FAR})$, that means R (the randomized algorithm for Fcs) should take at least $\Omega(\frac{R(\mathrm{FAR})}{t})$ many queries. Noticing that t is $\log n$, we finish the proof.

3 Assignment

Exercise 14. Show that $C(\mathrm{FAR}) = O(n^2)$.

Exercise 15. Try to finish the argument for the lower bound on $R(\mathrm{FAR})$. If it is not entirely clear feel free to look at the reference [1].

Exercise 16. Show that algorithm B is correct in the randomized sense.

Exercise 17. In the proof of lower bound of Fcs, show that $\sum_i (q_x)_i$ is less than the query complexity of R for all x.

References

1. Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*.
2. Shalev Ben-David. A super-grover separation between randomized and quantum query complexities. Preprint,.
3. Lov Grover. Fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, 06 1996.